B. AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

Claim 1 (Currently Amended): A method for monitoring thread usage in a server system, comprising:

sending an ioctl call in blocking mode on a socket designated for listening for incoming client requests to a server communicatively connected to a network and passing said incoming client requests to one from among a plurality of threads waiting in a thread pool;

responsive to a TCP layer detecting said listen socket in blocking mode, monitoring a thread count of at least one of a number of incoming requests waiting to be processed and a number of said plurality of threads remaining idle in said thread pool over a sample period; and

responsive to said TCP layer detecting a thread usage event, returning said ioctl call back with said thread count, such that a number of threads in said thread pool is ~~may be~~ dynamically adjusted to handle said thread count.

Claim 2 (Original): The method according to claim 1 for monitoring thread usage wherein said monitoring a thread count further comprises setting a counter to monitor said number of incoming requests waiting to be processed over a particular number of TCP slow timer processing cycles.

Claim 3 (Original): The method according to claim 1 for monitoring thread usage wherein said monitoring a thread count further comprises monitoring a minimum number of said number of threads remaining idle over said sample period.

Claim 4 (Original): The method according to claim 1 for monitoring thread usage further comprising:

dynamically adjusting a number of active threads in said thread pool according to said thread count to handle a current load.

Claim 5 (Original): The method according to claim 1 for monitoring thread usage further comprising:

initiating a TCP slow timer cycle;

processing all of a plurality of sockets during said TCP slow timer cycle;

responsive to completing said processing of all of said plurality of sockets, comparing said thread count with a threshold;

responsive to said thread count exceeding said threshold, designating a thread usage event with said number of incoming requests waiting to be processed; and

responsive to said thread count equaling zero after said sample period, designating a thread usage event with said number of threads remaining idle.

Claim 6 (Original): A system for monitoring thread usage in a server system, comprising:

a server system communicatively connected to a network;

said server system further comprising:

a socket designated for listening for incoming client requests to said server system from said network and passing said incoming client requests to one from among a plurality of threads waiting in a thread pool;

application means for sending an ioctl call in blocking mode on said socket;

TCP timer processing means for monitoring a thread count of at least one of a number of incoming requests waiting to be processed and a number of said plurality of threads remaining idle in said thread pool over a sample period; and

TCP layer means for returning said ioctl call back with said count, responsive to detecting a thread usage event.

Claim 7 (Original): The system according to claim 6 for monitoring thread usage wherein said TCP timer processing means further comprises setting a counter to monitor said number of incoming requests waiting to be processed over a particular number of TCP slow timer processing cycles.

Claim 8 (Original): The system according to claim 6 for monitoring thread usage wherein said TCP timer processing means further comprises monitoring a minimum number of said number of threads remaining idle over said sample period.

Claim 9 (Original): The system according to claim 6 for monitoring thread usage, wherein said server system further comprises:

means for dynamically adjusting a number of active threads in said thread pool according to said thread count to handle a current load.

Claim 10 (Original): The system according to claim 6 for monitoring thread usage, wherein said server system further comprises:

means for initiating a TCP slow timer cycle;

means for processing all of a plurality of sockets during said TCP slow timer cycle;

means responsive to completing said processing of all of said plurality of sockets, for comparing said thread count with a threshold;

means responsive to said thread count exceeding said threshold, for designating a thread usage event with said number of incoming requests waiting to be processed; and

means responsive to said thread count equaling zero after said sample period, for designating a thread usage event with said number of threads remaining idle.

Claim 11 (Original): A computer program product for monitoring thread usage in a server system, comprising:

a recording medium;

means, recorded on said recording medium, for sending an ioctl call in blocking mode on a socket designated for listening for incoming client requests to a server communicatively connected to a network and passing said incoming client requests to one from among a plurality of threads waiting in a thread pool;

means, recorded on said recording medium, for monitoring a thread count of at least one of a number of incoming requests waiting to be processed and a number of said plurality of threads remaining idle in said thread pool over a sample period, responsive to a TCP layer detecting said listen socket in blocking mode; and

means, recorded on said recording medium, for returning said ioctl call back with said thread count, responsive to said TCP layer detecting a thread usage event.

Claim 12 (Original): The computer program product according to claim 11 for monitoring thread usage wherein said means for monitoring a thread count further comprises:

means, recorded on said recording medium, for setting a counter to monitor said number of incoming requests waiting to be processed over a particular number of TCP slow timer processing cycles.

Claim 13 (Original): The computer program product according to claim 11 for monitoring thread usage wherein said means for monitoring further comprises:

means, recorded on said recording medium, for monitoring a minimum number of said number of threads remaining idle over said sample period.

Claim 14 (Original): The computer program product according to claim 11 for monitoring thread usage further comprising:

means, recorded on said recording medium, for dynamically adjusting a number of active threads in said thread pool according to said thread count to handle a current load.

Claim 15 (Original): The computer program product according to claim 11 for monitoring thread usage further comprising:

means, recorded on said recording medium, for initiating a TCP slow timer cycle;

means, recorded on said recording medium, for processing all of a plurality of sockets during said TCP slow timer cycle;

means, recorded on said recording medium, for comparing said thread count with a threshold, responsive to completing said processing of all of said plurality of sockets;

means, recorded on said recording medium, for designating a thread usage event with said number of incoming requests waiting to be processed, responsive to said thread count exceeding said threshold; and

means, recorded on said recording medium, for designating a thread usage event with said number of threads remaining idle, responsive to said thread count equaling zero after said sample period.

Claim 16 (Currently Amended): A method for monitoring thread usage to dynamically adjust a number of active threads in a thread pool of a server system, comprising:

sending an ioctl call in blocking mode on a socket designated for listening for incoming client requests to a server system communicatively connected to a network and passing said incoming client requests to one from among a plurality of active threads waiting in a thread pool;

responsive to a TCP layer of said server system detecting a thread usage event, receiving said ioctl call back with a thread count of at least one of a number of incoming requests waiting to be processed and a number of said plurality of threads remaining idle in said thread pool over a sample period; and

dynamically adjusting said number of active threads in said thread pool according to said thread count, such that said server system dynamically adjusts said thread pool to handle a current load.

Claim 17 (Original): A system for monitoring thread usage to dynamically adjust a number of active threads in a thread pool of a server system, comprising:

a server system communicatively connected to a network;

said server system further comprising:

means for sending an ioctl call in blocking mode on a socket designated for listening for incoming client requests and passing said incoming client requests to one from among a plurality of active threads waiting in a thread pool;

means responsive to a TCP layer of said server system detecting a thread usage event, for receiving said ioctl call back with a thread count of at least one of a number of incoming requests waiting to be processed and a number of said plurality of threads remaining idle in said thread pool over a sample period; and

means for dynamically adjusting said number of active threads in said thread pool according to said thread count to adjust a capacity to handle new client requests.

Claim 18 (Original): A computer program product for monitoring thread usage to dynamically adjust a number of active threads in a thread pool of a server system, comprising:

a recording medium;

means, recorded on said recording medium, for sending an ioctl call in blocking mode on a socket designated for listening for incoming client requests to a server system communicatively connected to a network and passing said incoming client requests to one from among a plurality of active threads waiting in a thread pool;

means, recorded on said recording medium, for receiving said ioctl call back with a thread count of at least one of a number of incoming requests waiting to be processed and a number of said plurality of threads remaining idle in said thread pool over a sample period, responsive to a TCP layer of said server system detecting a thread usage event; and

means, recorded on said recording medium, for dynamically adjusting said number of active threads in said thread pool according to said thread count to adjust a capacity to handle new client requests.